FACULTY OF COMPUTATIONAL SCIENCE AND ENGINEERING

Software Lab Project

Computer Analysis of Marine Animal Trajectory

by

Akomoneh Raymond Anumaneh

218100122

Supervised by

Prof. Nikolai Kornev

Content

List of Tables

List of Figures

0. Introduction

This report is about the Visualization and characterization of marine animal trajectories. The Animals, Seals, were tracked with GPS devices and information relating to their location such as longitude and latitudes over time recorded. Parameters such as trajectories, mean speed, straightness index, Sinuosity and Fractal Dimension are of interest.

Totally, 27 Seals were tracked but this analysis is done on 4 seals (G17, G20, S11, & S12) selected based on tracking duration as shown by table 1.0. The total data points which corresponds to the number of GPS signals that were recorded are also represented in figure 1.0. Longer durations were of interest because they provide more details and hence more meaningful analysis.

It is worth mentioning that this report closely follows the work of [1] and most of the Methods and definitions are unedited while the Traja and Trajr documentations were also consulted.

Table 1.0 Selected Seals

Species	**Grey Seals**		**Harbor Seals**	
	G17	**G20**	**S11**	**S12**
Sex	Male	Female	Male	Male
Body mass (kg)	108	58	112	118
Tracking Duration (days)	179	189	53	78

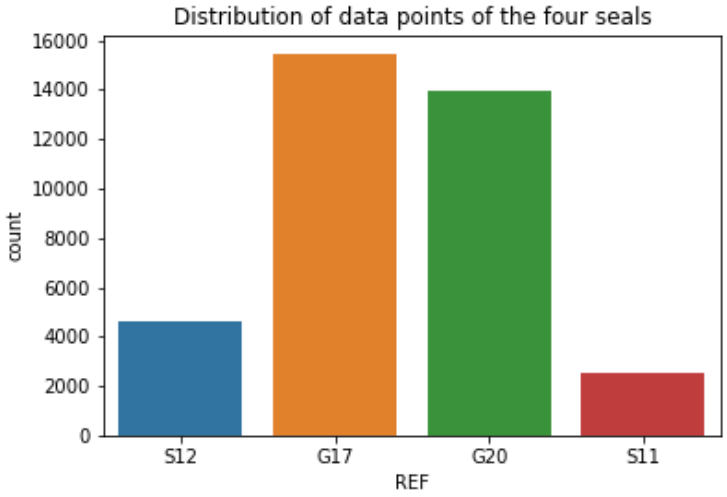


Figure 1.0: Distribution of Data points

CREATING TRAJECTORIES

1.0 Types of Trajectories

Basically, the trajectory of an animal is the curve described by the animal when it moves. The sampling of the trajectory implies a step of discretization, that is the division of this continuous curve into a number of discrete "steps" connecting successive relocations of the animal [2]. Two main classes of trajectories can be distinguished:

- **Trajectories of type I** are characterized by the fact that the time is not precisely known or not taken into account for the relocations of the trajectory;
- **Trajectories of type II** are characterized by the fact that time is known for each relocation.

This report follows type II trajectories.

1.1 Creating Trajectories with the R-package *Trajr*

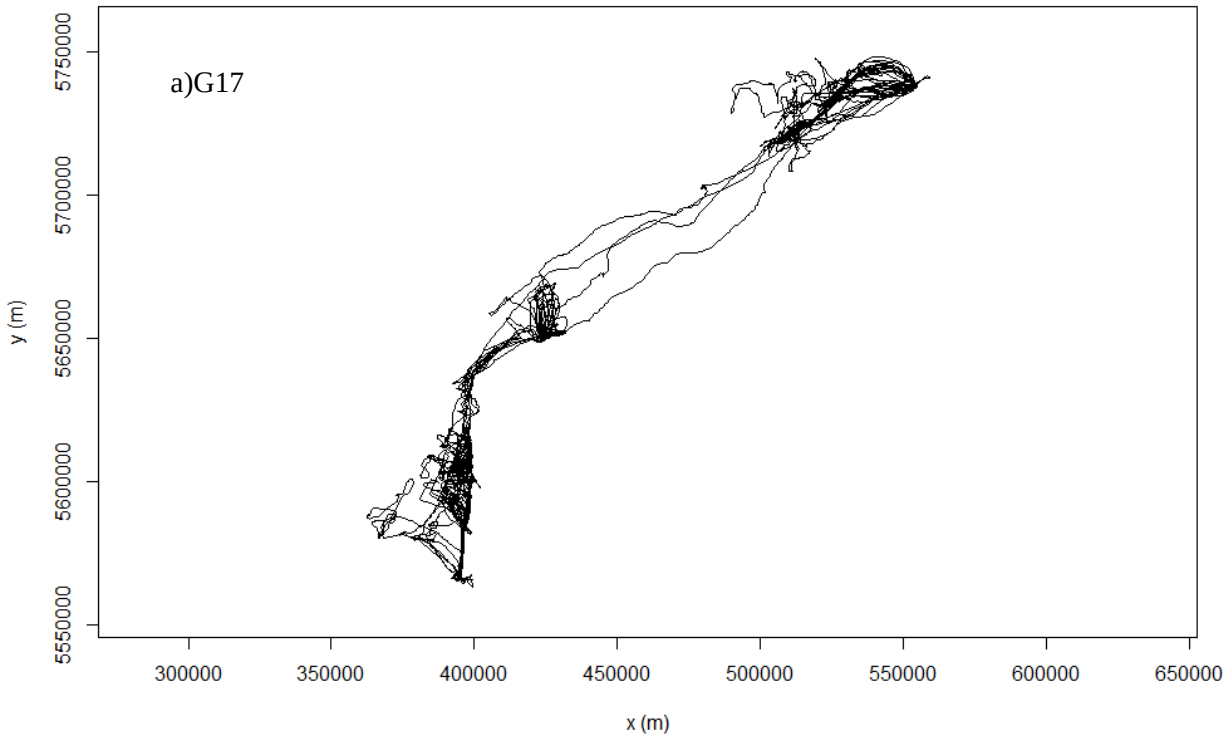
The trajectory coordinates are read from a data file such as a CSV file, and then passed in to the ***TrajFromCoords*** function of Trajr to create the trajectory as follows:

```
#Load Trajr Library
library("trajr")
#Read the coordinates into a DataFrame name "Coords_G17"
Coords_G17 = read.csv("mycoords_G17.csv")
#Create trajectory from coordinate and specify units
trajG17 = TrajFromCoords(coords_G17, xCol ="X", yCol =
"Y", timeCol = "time", spatialUnits = "m", timeUnits = "s"))
#plot trajectory
plot(trajG17)
```

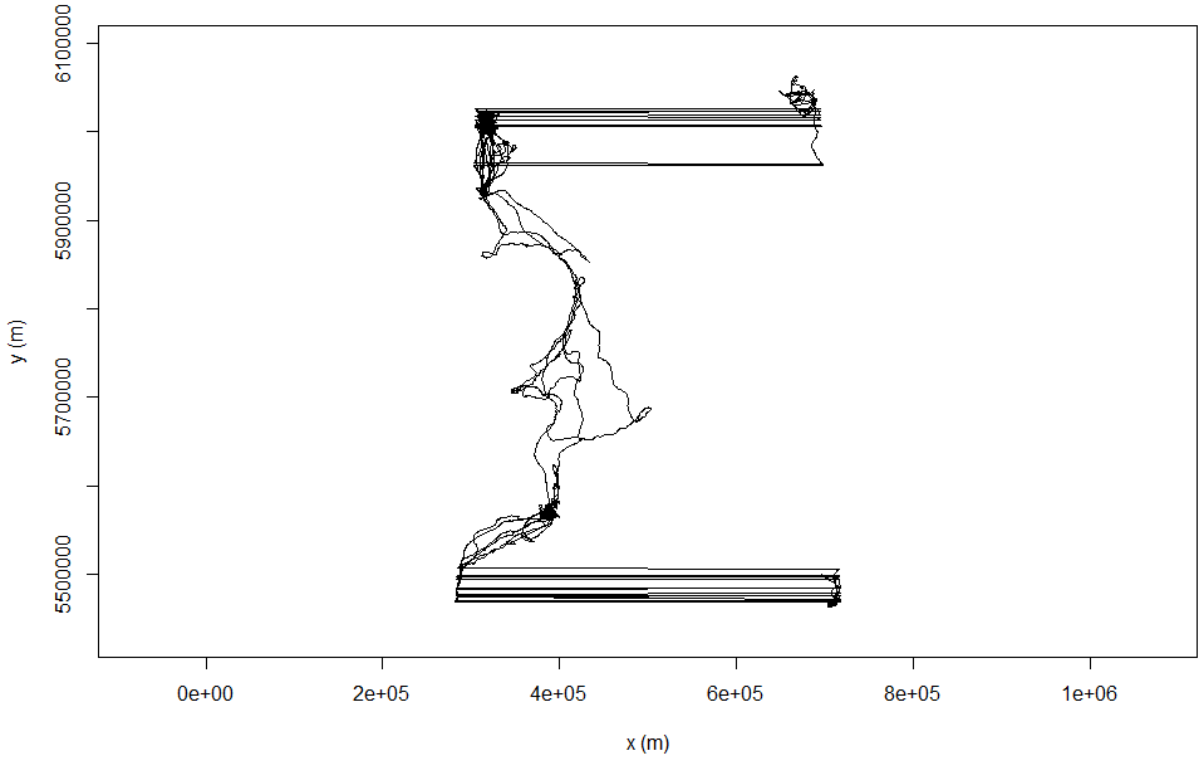
TrajFromCoords assumes that the first column in ***mycoords_G17*** contains **X** values, the second contains **Y** values, and the third column contains an optional time variable. This can be called by setting the ***timeCol*** argument of the trajectory to the respective time coordinate in ***Coords_G17***. ***Coords_G17*** could be a DataFrame or a series**.**

While trajectories have been modelled in various ways, trajr works with two theoretical models: *correlated random walks* in which the turning angle of each step is the direction of the previous step ± some error; and *directed walks*, or compass-based navigation, in which the angular errors at each step are added to the "ideal" or compass direction. Trajr generally assumes correlated random walks, but provides some support for directed walks [1].

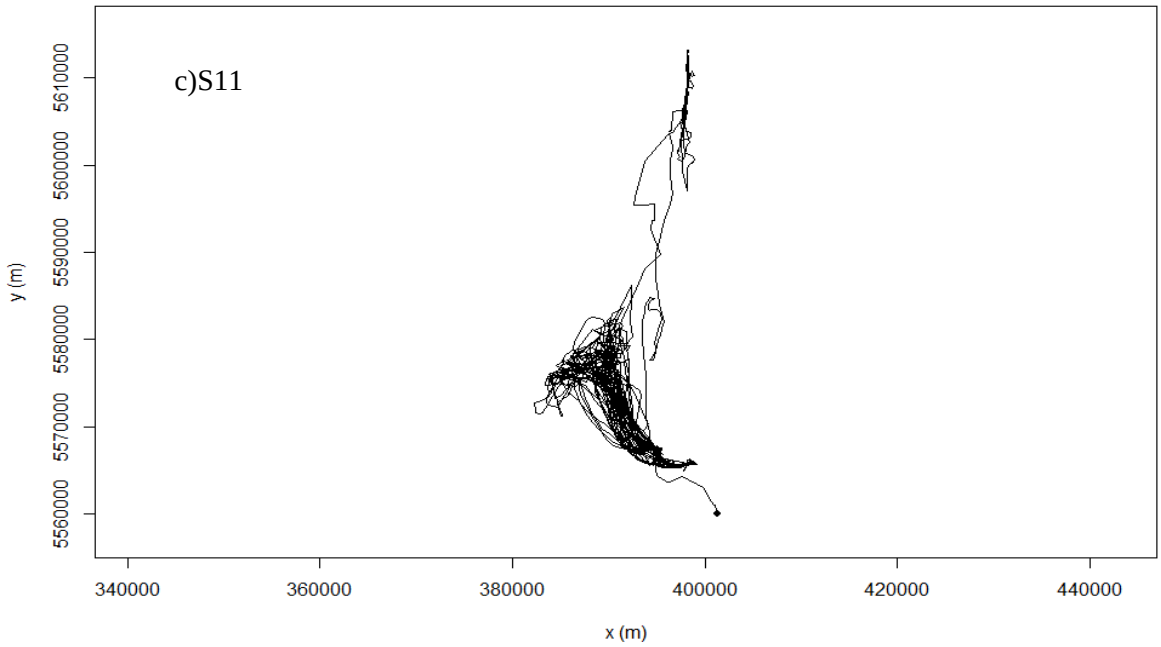
Trajectories produced with trajr are shown in Figure 1.1 a-d. *The horizontal lines on G20 results from the projection from latitude and longitude to Universal Transverse Mercator (UTM) XY and will greatly affect the results of Seal G20. (Refer to figure 1.2b for a corrected version).*



b)G20


y (m)
6100000
5900000
5700000
5500000
0e+00
2e+05
4e+05
6e+05
8e+05
1e+06
x (m)

c)S11


y (m)
5610000
5600000
5590000
5580000
5570000
5560000
340000
360000
380000
400000
420000
440000
x (m)

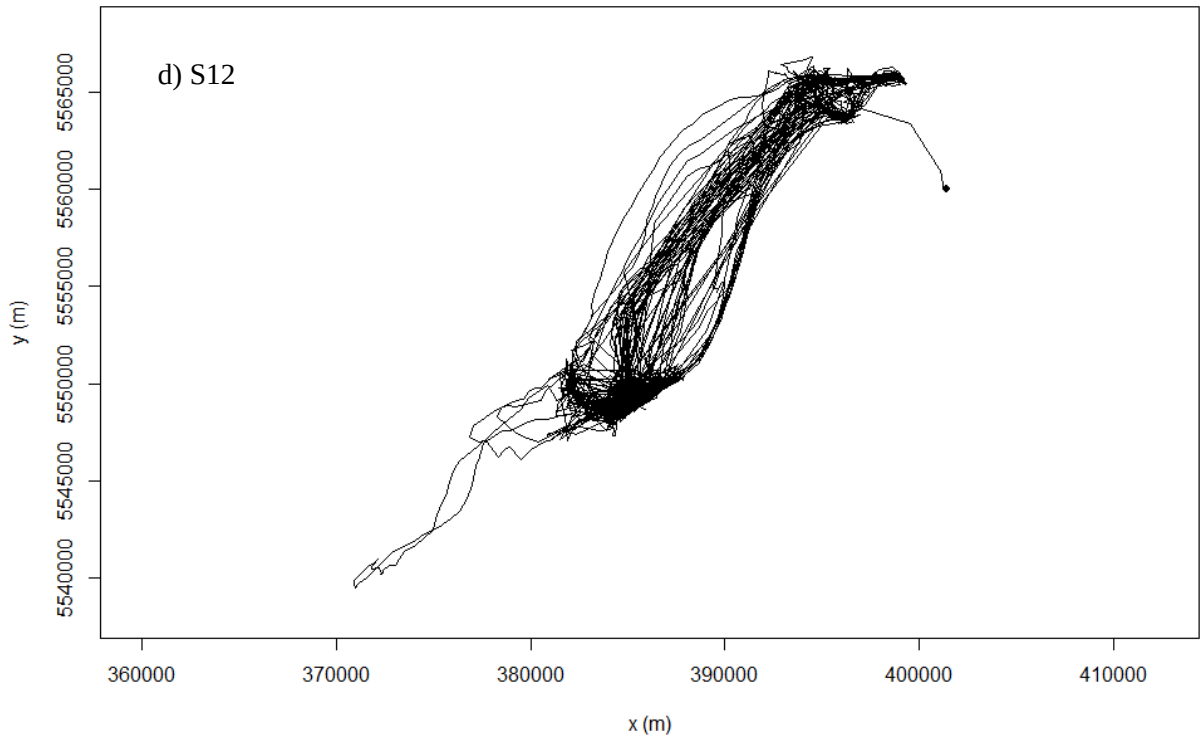


Figure 1.1 a-d : Type I Trajectories created with `trajr`.

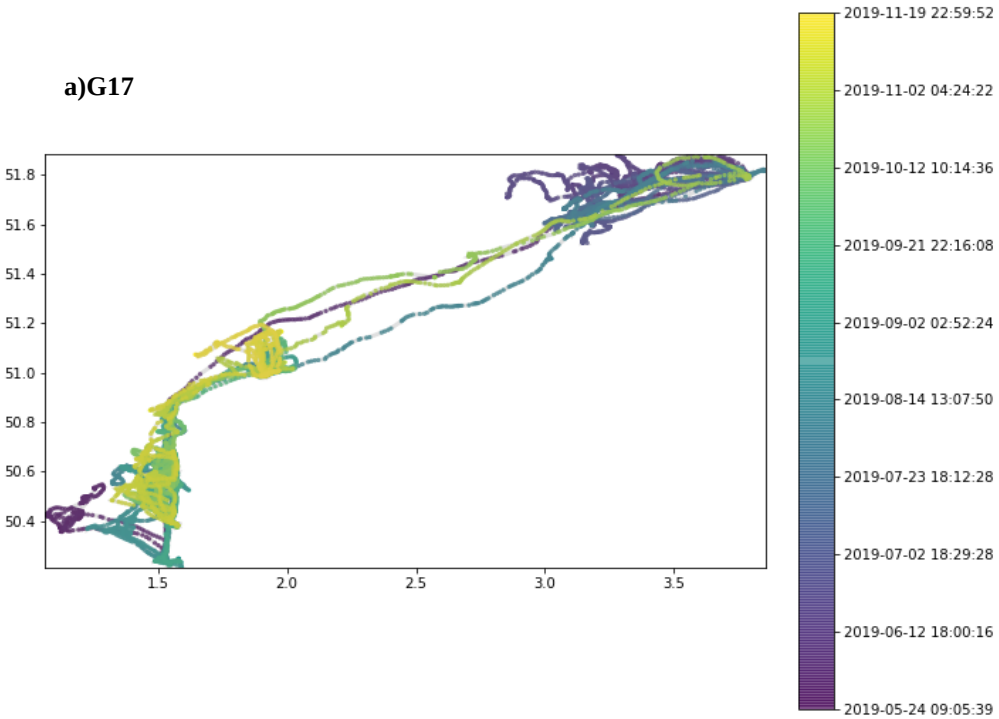
1.2 Creating Trajectories with *`Traja`*

Type II trajectories have been created using python's ***traja*** Library developed by Justin Shenk [3]. For the sake of this project, the ***x*** and ***y*** coordinates will be supplied with Longitudes and Latitudes respectively. Traja requires conversion in to a ***TrajaDataFrame*** and also takes a third optional time argument as follows:

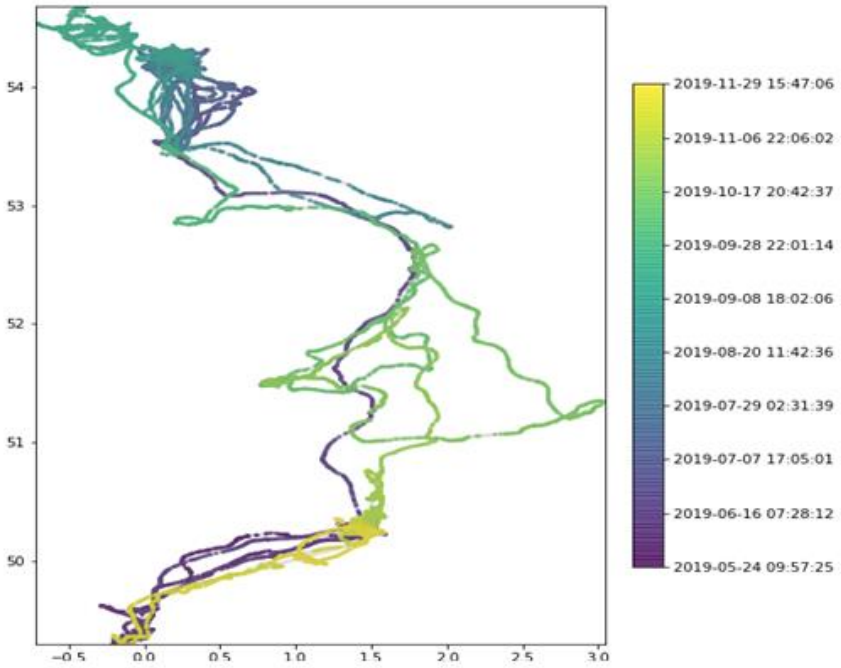
```
#import traja library
import traja

#Read Seal's data into a DataFrame
G17 = pd.read_csv("Seals_df.csv")

# Transform Seal DataFrame into a TrajaDataFrame
trjG17 = traja.TrajaDataFrame(G17)

# Plot trajectory by calling traja.plot()
figG17 = G17.traja.plot(figsize=[10.,11.])
```

The figure 1.2 a-d below shows the four trajectories created by `Traja`. The remarkable coloration shows the progress of the trajectory with time.

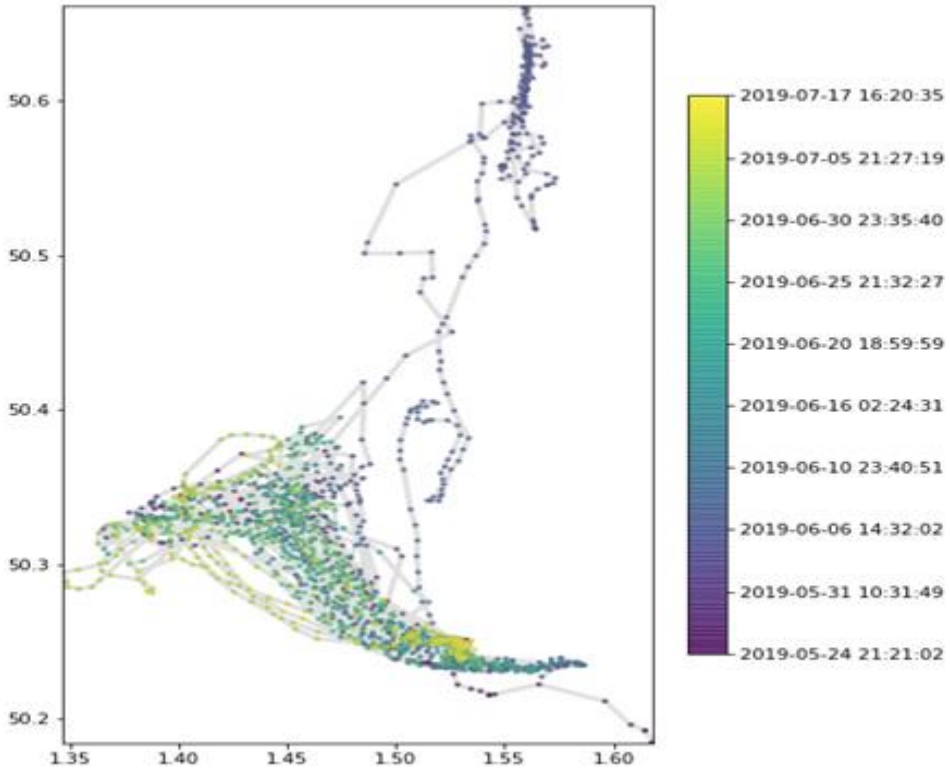
a)G17



b)G20


54
53
52
51
50
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
2019-11-29 15:47:06
2019-11-06 22:06:02
2019-10-17 20:42:37
2019-09-28 22:01:14
2019-09-08 18:02:06
2019-08-20 11:42:36
2019-07-29 02:31:39
2019-07-07 17:05:01
2019-06-16 07:28:12
2019-05-24 09:57:25

C) S11


50.6
50.5
50.4
50.3
50.2
1.35
1.40
1.45
1.50
1.55
1.60
2019-07-17 16:20:35
2019-07-05 21:27:19
2019-06-30 23:35:40
2019-06-25 21:32:27
2019-06-20 18:59:59
2019-06-16 02:24:31
2019-06-10 23:40:51
2019-06-06 14:32:02
2019-05-31 10:31:49
2019-05-24 21:21:02

d) S12

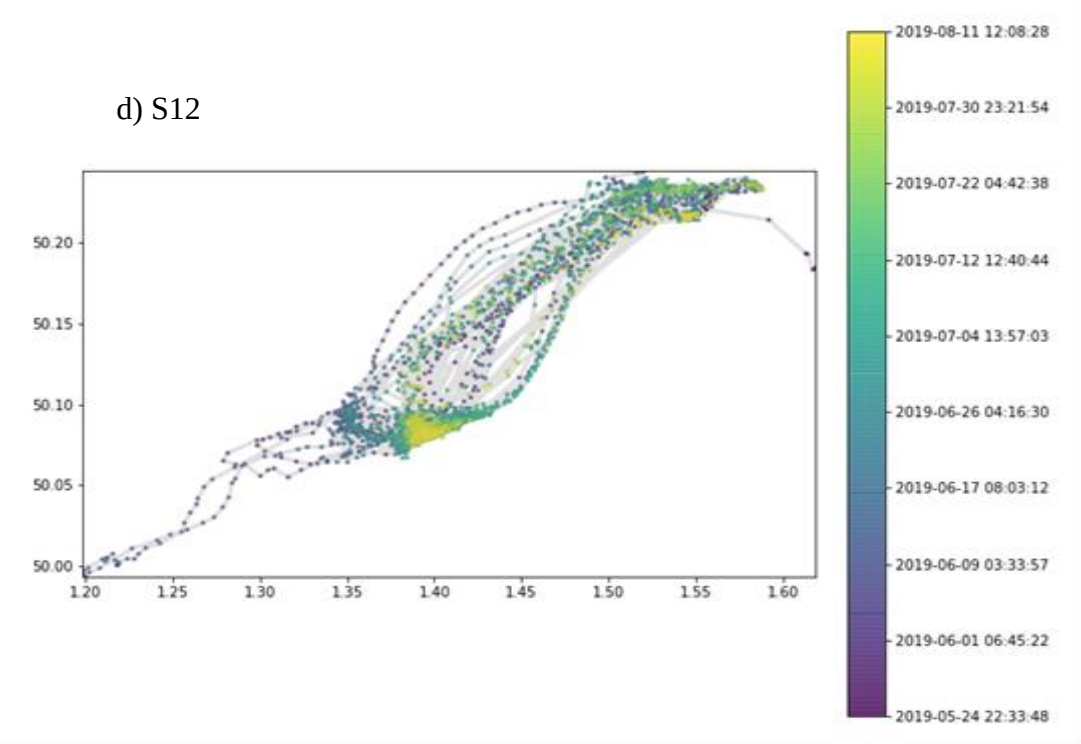


Figure 1.2 a-d : Type II Trajectories created with `traja`

1.3 **UTM and Transformation of Coordinates**

The Universal Transverse Mercator (UTM) is a system for assigning coordinates to locations on the surface of the Earth. Like the traditional method of latitude and longitude, it is a horizontal position representation, which means it ignores altitude and treats the earth as a perfect ellipsoid. However, it differs from global latitude/longitude in that it divides earth into 60 zones, each 6° of longitude in width and projects each to the plane as a basis for its coordinates. Specifying a location means specifying the zone and the *x*, *y* coordinate in that plane.

A Bidirectional UTM-WGS84 converter for python by [4] has been used to convert Longitudes and latitudes into the UTM XY coordinates required by both Traja and Trajr. The implementation in python as explained on the GitHub page (see references).

```
#import utm library
import utm
# Convert a (latitude, longitude) tuple into an UTM coordinate:
utm.from_latlon(51.2, 7.5)
>>> (395201.3103811303, 5673135.241182375, 32, 'U')
Syntax: utm.from_latlon(LATITUDE, LONGITUDE).
Return form: (EASTING, NORTHING, ZONE NUMBER, ZONE LETTER).
#Convert an UTM coordinate into a (Latitude, longitude) tuple:
utm.to_latlon(340000, 5710000, 32, 'U')
>>> (51.51852098408468, 6.693872395145327)
Syntax:  utm.to_latlon(EASTING, NORTHING, ZONE NUMBER, ZONE LETTER).
Return form: (LATITUDE, LONGITUDE).
#Define functions to get X and Y coordinates from utm
def get_X(row):
return utm.from_latlon(row['LAT'], row['LON'])[0]
def get_Y(row):
return utm.from_latlon(row['LAT'], row['LON'])[1]
#get X and Y coordinates and store as new columns in the df
Seal_G17['X'] = Seal_G17.apply(get_X, axis=1)
Seal_G17['Y'] = Seal_G17.apply(get_Y, axis=1)
```

1.4 Basic *trajr* parameters

Parameter	Description
TrajGetNCoords	Returns the number of coordinates of a trajectory
TrajLength	Returns the total length of the trajectory
TrajDistance	Returns the straight-line distance from the start to the end of the trajectory
TrajDuration	Returns the duration of the trajectory
TrajMeanVelocity	Returns the mean velocity vector of the trajectory
TrajMeanVectorOfTurningAngles	Returns the mean vector of the turning angles
TrajExpectedSquareDisplacement	Returns the expected square displacement of a correlated random walk

Table 1.1: Describing Basic *trajr* parameters

1.5 *Trajr* parameters applied to the four Seals

Parameter	G17	G20	S11	S12
TrajGetNCoords	15,452	13,935	2,537	4,645
TrajLength	5,455,885	15,070,486	914,083.5	1,675,010
TrajDistance	88,938.18	10,386.58	9,799.389	7,083.866
TrajDuration	15,515,653	16,350,581	4,647,573	6,788,080
TrajMeanVelocity	0.001929153 +0.005397778i	4.869261e-04- 4.079656e-04i	-0.001315256 +0.001647986i	-5.491435e-04 +8.874056e-04i
TrajMeanVectorOf TurningAngles	0.9669657 +0.0023647i	0.9618576 -0.0027684i	0.9493727 +0.0057143i	0.949455 +0.0133539i
TrajExpectedSquare Displacement	115,306,160,874	928,139,593,935	12,601,698,585	22,036,449,920

Table 1.2: *Trajr* parameters applied to the four Seals

2.0 TRAJECTORY ANALYSIS

Analysis may be divided into measures of speed or acceleration, and measures of straightness or tortuosity.

2.1 Speed and Acceleration

The ***TrajDerivatives*** function calculates linear speed and acceleration along a Trajectory. Traja recommends smoothing of noisy trajectories before the derivatives are calculated. The results show that the average speed across the four seals is about 0.4m/s.

Description	**G17**	**G20**	**S11**	**S12**
Average speed (m/s)	0.4421209	1.261982	0.379479	0.3957427
Minimum speed (m/s)	2.659274e-05	0.00058100	0.001570785	0.0005393309
Maximum speed (m/s)	2.505583	40.101260	2.216285	2.122943
Standard Deviation of speed	0.4185635	3.226617	0.3551129	0.3965464
Average Acceleration (m/s^2)	3.63980e-05	0.0001816152	5.339235e-05	6.325358e-05

Table 2.0: Trajectory measures of speed and acceleration *(Values of Seal G17 in red are suspiciously incorrect. This results from the erroneous projection from latitude and longitude into utm XY coordinates)*

2.2 Straightness

The concept of tortuosity (or conversely, straightness) is intuitively an important characteristic of trajectories that is inversely related to the efficiency of navigation towards a destination and controls local searching intensity [5].

This characteristic is also valuable in locomotor mimicry studies, allowing verification of whether a mimic displays a tortuous path to resemble its model, even though moving in a straight line would have allowed it to reach its destination faster.

Various methods to measure the straightness of trajectories exist and some are available within trajr. The simplest is D/L, where D is the beeline distance between the first and last points in the trajectory, and L is the path length travelled [6]. This straightness index is calculated by the Trajr function ***TrajStraightness***, and is a number ranging from 0 to 1, where 1 indicates a straight line. [5] considers the straightness index to be a reliable measure of the efficiency of a directed walk, but inapplicable to random trajectories. [6] considers this straightness index to be an approximation of r, which is the length of the mean vector of turning angles after discretizing to a constant step length. r can be calculated by calling ***Mod(TrajMeanVectorOfTurningAngles(trj)), assuming trj*** is a Trajectory with constant step length. The results presented in table 2.0 shows that none of the trajectories of the four seals is straight.

Seal	**Straightness Index**
G17	0.01630133000
G20	0.00003045027
S11	0.00593888400
S17	0.00242494100

Table 2.0: Trajectories measures of straightness

2.3 Sinuosity

The sinuosity index defined by [5] may be an appropriate measure of the tortuosity of a random search path. Sinuosity is a function of the mean cosine of turning angles, and is a corrected form of the original sinuosity index defined by [7].

The function ***TrajSinuosity2*** calculates the corrected form while ***TrajSinuosity*** calculates the original index. Sinuosity is calculated for a trajectory with a constant step length, so trajectories may first require rediscretization. In the absence of a biologically meaningful step size, discretizing to the mean step length of the trajectory will produce a trajectory with approximately the same shape and number of steps as the original [1].

Seal	**Sinuosity1**	**Sinuosity2**
G17	0.01845154	0.01370951
G20	0.03585707	0.004137565
S11	0.06724543	0.04118568
S17	0.02343371	0.01682016

Table 2.1: sinuosity indices

With remarkable improvement due to the correction introduced by Sinuosity2, both values are low. This means that the cosine of the turning angles is small and since the cosine of the turning angles is small, it in turn means that the turning angles are closer to 90 degrees. However, more details can be revealed if the timing is reduced from an average of 10minutes to 10 seconds. It is possible to obtain completely different results with such changes.

2.4 E-max

Emax, the maximum expected displacement, is a single-valued measure of straightness defined by [8]. Emax-a is a dimensionless, scale-independent measure of the maximum possible expected displacement. Emax-b is Emax-a * mean step length, and gives the maximum possible expected displacement in spatial units [1]. Values closer to 0 are more sinuous, while larger values (approaching infinity) are straighter.

Seal	E-max Unsmoothed	E-max smoothed	Emax-b smoothed
G17	1.044200	29.27161	10,336.07
G20	1.345194	25.21757	27,274.37
S11	0.6261382	18.75218	6,759.093
S17	0.4805549	18.78435	6,775.189

Table 2.2: Emax Values

3.0 FRACTAL DIMENSION CALCULATION

3.1 Defining Fractal Dimension

Fractal dimension has been considered a promising measure of straightness or tortuosity, varying between 1 (*straight line movement*) and 2 (*Brownian motion*). However, several studies have found it inappropriate for use with animal trajectories, as animal trajectories are not fractal curves, and the fractal dimension of a non-fractal curve depends critically on the range of step sizes used to calculate it [2], [9]. Nonetheless, fractal dimension continues to be used, and the ***TrajFractalDimension*** function, by default, calculates fractal dimension using the modified dividers method to account for truncation error [9]. This method is rather too slow and the results herein presented were obtained by running this function on the entire Data of each trajectory.

Some of the arguments of this function include: Trj: the trajectory to calculate fractal dimension for, stepSizes: a vector of step sizes (aka divider sizes) used to calculate path lengths, adjustD:[TRUE or FALSE]. If TRUE, path length is adjusted for truncation error [9]. And dMean:[TRUE or FALSE]. If TRUE, the fractal dimension is calculated starting from the beginning of the trajectory, then re-calculated starting from the end and moving backwards. The value returned is the mean of the two fractal dimensions [9], [10].

The "stepSizes" argument, that affects the results to a large extent has been chosen to coincide with the overall mean step sizes for all four trajectories in this primary calculation. This was then applied independently across each trajectory. The results presented on table 3.0 below shows that all the seals being investigated follow a *straight line movement*. However, a close look at a stochastic section of trajectory G17 (in sections 3.1 - 3.3) reveals some interesting information on the Fractal Dimension of G17.

Table 3.0: Fractal Dimension Values

Seal	Fractal Dimension
G17	1.119366
G20	1.049446
S11	1.113465
S12	1.119362

3.2 Testing For a Fractal Curve

As described in trajr documentation [10], it is possible to test whether a trajectory is a fractal curve for a range of step sizes using the *TrajFractalDimensionValues* function. If the plotted points have a tendency to lie on a straight line (see figure 3.1 below), then the trajectory is a fractal curve for that range of step sizes. However, according to same document, typical trajectories will result in a curve rather than a straight line. The example code provided in [10] demonstrates how to plot path length for a range of step sizes. Figure 3.1 below is the result obtained by applying this test to a stochastic section of trajectory G17 shown in figure 3.0. The stepSizes argument was varied to get a plot that was almost linear from where the final step size was selected and used to compute the Fractal Dimension. The code that follows that a Fractal Dimension Value of about 1.5 could be obtained by picking a range of Stepsizes from the "linear" section of figure 3.1. One way to get Fractal Dimension from this plot is to find a straight line that connects most of the points in the linear region and get the slope of the line but a different much simpler approach will be used in the following section.

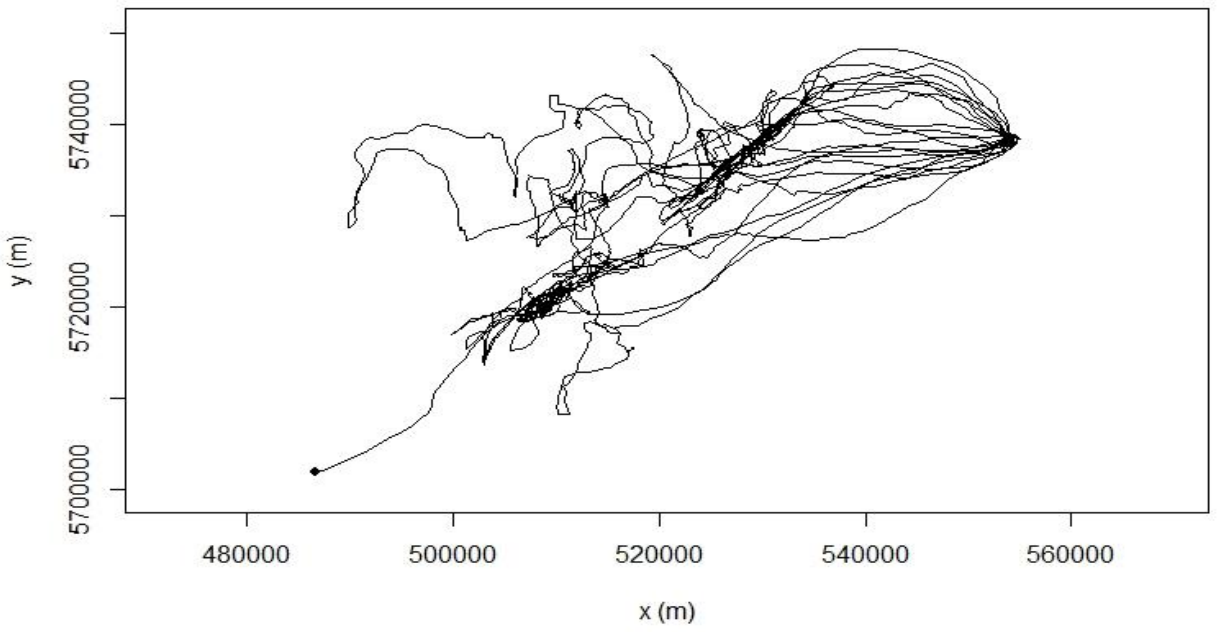


Figure 3.0: Plot of a stochastic section of the trajectory G17

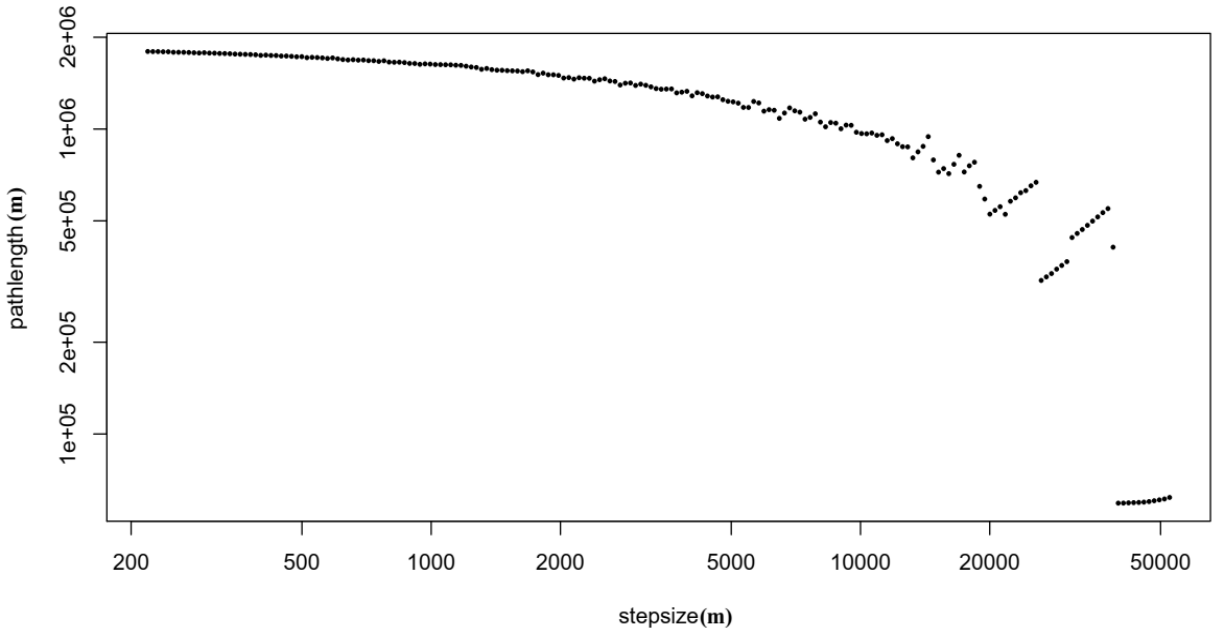


Figure 3.1: Plot of Pathlength against stepsize

```
#get Trajectory from CSV

df_G17 = read.csv("Seal_G17.csv")

#Select subset of the Trajectory

G17 = df_G17[680:5000,]

G17_df = G17[, c("X", "Y", "time.sec_cumsum")]

#consntruct trajectory from Coordinates

trajrG17 = TrajFromCoords(G17_df, xCol ="X", yCol = "Y", timeCol = "time.sec_cumsum",
spatialUnits = "m", timeUnits = "s")

plot(trajrG17)

#Trial program to plot and compute Fractal Dimension of a section of the G17 Trajectory

set.seed(42)

meanlength <- mean(TrajStepLengths(trajrG17))

# Use 200 step sizes from 2*mean step length to 120*mean step length.

stepSizes <- TrajLogSequence(2*meanlength, 120 * meanlength, 200)

#Plot TrajFractalDimensionValues to determine if trajectory is fractal and choose
meaningful stepsizes from the linear plot

plot(TrajFractalDimensionValues(trajrG17, stepSizes), log="xy", pch=16, cex=.5)

#Compute Fractal Dimension for the combination of stepsizes the reults to a straight line

D = TrajFractalDimension(trajrG17, stepSizes, adjustD = TRUE, dMean = TRUE)

D = 1.570523
```

3.3 Variation of Fractal Dimension with Step sizes

In order to understand the relationship between Fractal Dimension and Step sizes, these two quantities have been plotted in figure 3.3 below.

A range of step sizes were picked from the "linear" region of figure 3.1. and assigned to a vector called sizes. FractalDimension is initialized to a Null vector and a for loop is used to compute its value using the **`TrajFractalDimension`** algorithm from Trajr library (see code below) for each step size. A plot is made of these two quantities as shown in figure 3.2. Again this is not straight line because of the errors incurred as the step size approaches zero. A straight line is imposed on this plot using the red dotted lines. The Fractal Dimension value is the intersection of this line with the FractalDimension axis. The value obtained is almost equal to the one obtained through computation with the **`TrajFractalDimension`** function**.**

```
# Vector of Step sizes

sizes = c(200, 300, 450, 500, 1000, 1500, 2000, 3000, 4000, 5000, 10000, 15000,
20000)

# Assign FractalDimension to a NULL vector

FractalDimension = c()

#for loop: using each step size from sizes, compute FractalDimension and append
results to FractalDimension vector assigned above.

for (i in sizes) {

  stepSizes <- TrajLogSequence(i, 120*450, 200)

  Fractal <- TrajFractalDimension(trajrG17, stepSizes, adjustD = TRUE, dMean =
TRUE)

  FractalDimension <- append(FractalDimension, Fractal)

}

# Make a plot of FractalDimension against the stepsizes

plot(sizes, FractalDimension, type="l", xlab="Step sizes (m)",
ylab="FractalDimension",

     main="Variation of FractalDimension  with Stepsize")
```

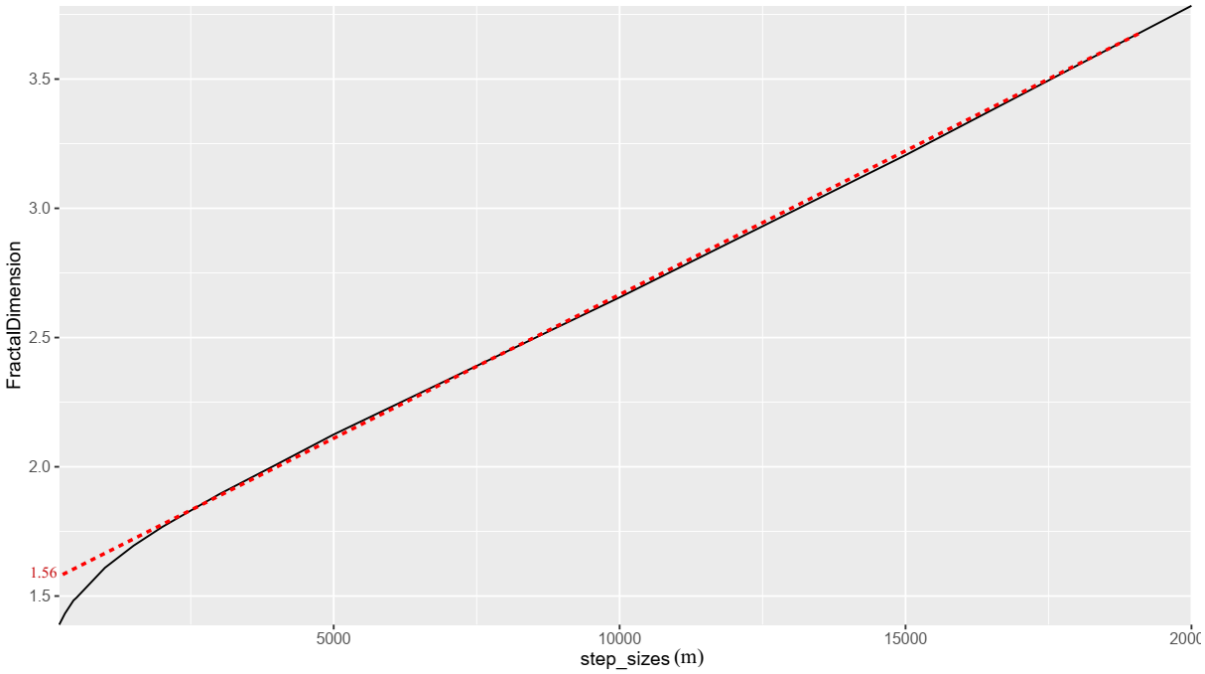


Figure 3.2: Variation of Fractal Dimension with Stepsizes

4.0 Conclusion

This report has presented the analysis and visualization of the trajectories of four Seals. Throughout the analysis, results obtained have been presented in a way that permits a direct comparism of the key aspects being examined.

The average speed across the four seals was found to be about 0.4m/s. None of the trajectories was straight. The so-called "biologically meaningful" step size, was interpreted to be the mean step length of the four trajectories.

Corrections introduced by Sinuosity2 lead to improved results compared to Sinuosity1 whereas both values were found to be low. Since the values were low, the cosine of the turning angles were small, meaning that the turning angles were closer to 90 degrees. However, these results can be challenged if the average timing in between steps is reduced from 10 minutes to 10 seconds.

The stochastic section of trajectory G17 passed the Fractal test and this test provided firsthand information that led to the computation of it Fractal Dimension. The Fractal Dimension value obtained through computation using **`TrajFractalDimension`** function and meaningful step size and the final value obtained by varying the Fractal Dimension with step size were almost equal. Therefore one may serve as a verification to the other.

Through out this project, Fractal Dimension was computed using the modified dividers method that minimizes truncation error. This method is however too slow. For future works, similar methods such as the Box-Counting method could be used and the results compared.

References

[1] . D. McLean and V. M. Skowron , "trajr: An R package for Characterisation of Animal Trajectories," *Ethology,* pp. 124:440-448, 2018.

[2] P. Turchin, "Fractal Analyses of Animal Movement: A Critique," *Ecology,* no. 77(7), pp. 2086-2090, 1996.

[3] J. Shenk, " A Python Trajectory Analysis Library pypi.org," 2009. [Online]. Available: https://pypi.org/project/traja/. [Accessed 22 08 2020].

[4] B. V. Tobias B., V. A. Bart and I. B. Torstein , "Bidirectional UTM-WGS84 converter for python," 2012. [Online]. Available: https://github.com/Turbo87/utm. [Accessed 11 8 2020].

[5] S. Benhamou, "How to reliably estimate the tortuosity of an animal's path," *Journal of Theoretical Biology,* vol. 2, no. 229, pp. 209-220, 2004.

[6] E. Batschelet, "Circular statistics in biology," *ACADEMIC PRESS,* p. 388, 1981.

[7] P. Bovet and Benhamou, S., "Spatial analysis of animals' movements using a correlated random walk model," *Journal of Theoretical Biology,* vol. 4, no. 131, pp. 419-433, 1988.

[8] Cheung, A., Zhang, S., Stricker, C. and Srinivasan, M. V., "Animal navigation: the difficulty of moving in a straight line," *Biological Cybernetics,* vol. 1, no. 97, pp. 47-61, 2007.

[9] N. VO., " Improving accuracy and precision in estimating fractal dimension of animal," *Acta Biotheor,* vol. 1, no. 54, pp. 1-11, 2006.

[10] J. McLean, "Package 'trajr': Animal Trajectory Analysis," 2019.

[11] Veronica D. and Erik D., "Fractal Analysis Can Explain Individual Variation in Dispersal Search Paths," *Wiley,* vol. 85, no. 5, pp. 1428-1438, 2004.

[12] B. Mandelbrot, "How long is the coast of Britain? Statistical selfsimilarity and fractional dimension," *Science,* vol. 156, no. 3775, p. 636–638, 1967.